Towards Autonomic Science Infrastructure: Architecture, Limitations, and Open Issues

Rajkumar Kettimuthu
Zhengchun Liu
Ian Foster
Peter H. Beckman
{kettimut,zhengchun.liu,foster,beckman}@anl.gov
Argonne National Laboratory
Lemont, IL, USA

Alex Sim
Kesheng Wu
{asim,kwu}@lbl.gov
Lawrence Berkeley National Laboratory
Berkeley, CA, USA

Wei-keng Liao
Qiao Kang
Ankit Agrawal
Alok Choudhary
{wkliao,qiao.kang,ankitag,choudhar}@eecs.northwestern.edu
Northwestern University
Evanston, IL, USA

ABSTRACT

Scientific computing systems are becoming increasingly complex and indeed are close to reaching a critical limit in manageability when using current human-in-the-loop techniques. In order to address this problem, autonomic, goal-driven management actions based on machine learning must be applied end to end across the scientific computing landscape. Even though researchers proposed architectures and design choices for autonomic computing systems more than a decade ago, practical realization of such systems has been limited, especially in scientific computing environments. Growing interest and recent developments in machine learning have spurred proposals to apply machine learning for goal-based optimization of computing systems in an autonomous fashion. We review recent work that uses machine learning algorithms to improve computer system performance, identify gaps and open issues. We propose a hierarchical architecture that builds on the earlier proposals for autonomic computing systems to realize an autonomous science infrastructure.

CCS CONCEPTS

• **Computer systems organization → Architectures**; • **General and reference → Cross-computing tools and techniques**;

KEYWORDS

Autonomic Management System; Autonomic Science Infrastructure

ACM Reference Format:
Rajkumar Kettimuthu, Zhengchun Liu, Ian Foster, Peter H. Beckman, Alex Sim, Kesheng Wu, Wei-keng Liao, Qiao Kang, Ankit Agrawal, and Alok Choudhary. 2018. Towards Autonomic Science Infrastructure: Architecture, Limitations, and Open Issues. In *AI-Science'18: Autonomous Infrastructure for Science, June 11, 2018, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3217197.3217205

AI-Science'18, June 11, 2018, Tempe, AZ, USA

ACM ISBN 978-1-4503-5862-0/18/06…$15.00
https://doi.org/10.1145/3217197.3217205

1 INTRODUCTION

Scientists frequently develop code and analyze data on laptops or workstations, leveraging scientific software toolkits, and then run large simulations on leadership-class systems. The HPC center was previously the nexus of the scientific computing universe, both administratively and computationally. Users brought their codes and their data to computing facilities, and then operations teams configured and monitored HPC systems to achieve required uptimes and queue wait times, often by applying heuristics or experimentation to tune parameters viewed as important for system performance. Traditional low-level systems (storage systems, computer networks, operating systems) do not make extensive use of machine learning. In most real-world use cases, data do not perfectly follow any known pattern, and the engineering effort to build specialized solutions for every use case is usually too high. Currently, computer systems are filled with heuristics for performance tuning that usually involves numerous iterations of benchmark cycles that are slow and costly. For example, heuristics are used in compilers for instruction scheduling, register allocation, and loop nest parallelization strategies [52]; in networking for TCP window size and pacing decisions, backoff for retransmits, and data compression [3]; and in operating systems for process scheduling, buffer cache insertion/replacement, and file system prefetching [67]. However, since distributed teams and complex workflows now span resources from telescopes and light sources to fast networks, a single, centralized administrative team and software stack that relies on heuristic configuration cannot coordinate and manage all the resources. Each facility has its own goal(s) and constraint(s). Resources must begin to respond autonomically to meet facility goals, repairing and tuning their behavior while servicing scientific workflows.

The first decade of the 21st century saw considerable work on architectures and design choices for autonomic computing systems [2, 16, 27, 28, 57, 73]. But the practical realization of such systems has been limited, especially in scientific computing environments. Recent years have seen a rapid growth of interest in exploiting monitoring data and applying machine learning for automated management and performance analysis [8]. Such methods have been applied in HPC for purposes including compute node allocation, job scheduling, and power efficiency [7]; in data transfer nodes to determine optimal parameters so as to maximize performance [42]; in job schedulers to determine which tasks/VMs to colocate and which tasks to preempt [50, 60]; in processor chip

design for physical circuit layout and test case selection [10]; in automatic synthesis of specialized index structures, termed learned indexes, with low engineering cost [32]; and in parallel distributed file systems for file striping and avoidance of peer contention [36].

These examples have demonstrated that machine learning models can provide significant benefits over heuristic approaches in specific situations and for specific system configuration problems. Might many such models (with possibly conflicting goals) now be combined to create an autonomous science infrastructure capable of optimizing across many different configuration choices in order to achieve higher-level goals? Building such a system is perhaps analogous to the challenge of creating a self-driving car that must simultaneously navigate, avoid accidents, obey traffic laws, ensure passenger comfort, avoid overloading vehicle components, and conserve fuel—although with many more configuration choices and much less training data. It is a challenging research problem. In this paper we address this problem in three ways:

- Review the state of the art in the application of machine learning to computer system optimization
- Summarize open issues and opportunities
- Propose a goal-based hierarchical architecture for autonomous science infrastructure

2 STATE OF THE ART

In this section we review literature relevant to autonomous computing systems. First we review the literature on architecture and general concepts for autonomous computing systems. Then we review the literature on autonomous capabilities for key elements in scientific computing environments, namely, high-performance computers, storage, network, data transfer nodes, and scientific instruments.

2.1 Architecture and general concepts

In 2001 IBM released a manifesto observing that the main obstacle to further progress in the IT industry is a looming software complexity crisis. The manifesto pointed out that the difficulty of managing today's computing systems goes well beyond the administration of individual software environments [29]. In 2003, a special issue of the IBM Systems Journal [16] explored a broad set of ideas and approaches to autonomic computing, presenting some first steps toward creating self-managed computing systems. And in a 2003 article citing the IBM manifesto, Kephart and Chess [29] noted that autonomic computing, while "perhaps the most attractive approach to solving this problem," also is a "grand challenge that reaches far beyond a single organization . . . [whose] realization will take a concerted, long-term, worldwide effort by researchers in a diversity of fields." In 2004, White et al. [72] described an architectural approach to achieve the goals of autonomic computing. Their architecture communicated interfaces and behavioral requirements for individual system components; described how interactions among components are established; and recommended design patterns for self-configuration, self-optimization, self-healing, and self-protection. Many of these ideas were validated in two prototype autonomic computing systems. From another perspective, Tesauro et al. [69] presented a decentralized architecture for autonomic computing based on multiple interacting agents called autonomic elements. They created a prototype showing how a collection of elements self-assembles, recovers from certain classes of faults, and manages the use of computational resources in a dynamic multiapplication environment.

In 2005, Parashar et al. [57] surveyed related work and presented an introduction to autonomic computing, its challenges, and opportunities. Salehie et al. [63] proposed a categorization of complexity in IT systems and presented an overview of autonomic computing research area. They summarized the major autonomic computing systems that had been developed and outlined the research issues and challenges. In 2006, IBM published its 4th edition of an architectural blueprint for autonomic computing [25]. In 2007, Nami et al. [53] provided a thorough survey of autonomic computing. Based on the survey, the authors claimed that autonomic computing still is a new concept in large-scale heterogeneous systems with many challenges and issues. Tesauro [68] used reinforcement learning for autonomic computing. The case studies showed that standard online reinforcement learning can learn effective policies in feasible training times. In 2008, Huebscher et al. [23] described research that had been seen as seminal in influencing a large proportion of early work on autonomic computing. They claimed that autonomicity is not a well-defined subject and that different systems adhere to different degrees of autonomicity. They concluded that although autonomic computing has become increasingly interesting and popular, it remains a relatively immature topic. From another perspective, Litoiu et al. [38] tried to address this problem by porting methods from adaptive control theory. They examined the control theory approach, explained several control strategies with examples from both classical control theory and self-adaptive system domains, and showed how the issues addressed by these strategies can and should be seriously considered for autonomous systems.

2.2 High-performance computers

Here we discuss work on autonomous modules for compute clusters. We group this work into three categories: scheduling and resource management, failure prediction (especially for large supercomputers), and power efficiency.

Scheduling and resource management. Dynamic scheduling of tasks in large-scale HPC platforms usually is accomplished by using heuristics based on task characteristics and backfilling strategy. Defining heuristics that work efficiently in different scenarios is difficult, however, especially when considering a large variety of task types and platform architectures. Machine learning has been used to predict job execution time [17, 47] and to predict resource demands [12, 21, 31] and job arrival time [4, 56]. Carastan-Santos et al. [7] presented a methodology based on simulation and machine learning to obtain dynamic scheduling policies. Specifically, by using simulations and a workload generation model, they determined the characteristics of tasks that lead to a reduction in the mean slowdown of tasks in an execution queue. They then used a nonlinear function to model these characteristics and applied the function to select the next task to execute in a queue. Their approach resulted in performance improvements when the nonlinear function was applied to real workload traces from highly different machines, which proved the generalization capability of the obtained heuristics.

Stragglers, which are uncommon within a single job, are exceptionally slow tasks within a job that significantly delay its completion. Systematically and rigorously identifying their root causes has promising potential for optimizing the scheduler so that it avoids machines predicted to perform poorly for a given task [60, 74]. Zheng et al. [75] proposed a statistical machine learning framework to automatically identify recurring causes of stragglers and concisely reveal sophisticated causes and provide domain insight from traces of datacenter-scale jobs. The framework offers interpretability, reliability, and scalability.

Efficient resource allocation, considering various constraints and goals, is a fundamental requirement in HPC systems [24]. It plays a vital role in performance improvement and user satisfaction with a computer system.

Currently, a heterogeneous distributed environment with a mixture of hardware devices such as CPUs and GPUs is commonly used for training and inference with neural networks [50]. Mirhoseini et al. [50] proposed a method that learns to optimize device placement for TensorFlow [1] computational graphs. The key aspect of their method is the use of a sequence-to-sequence machine learning model to predict which subsets of operations in a TensorFlow graph should run on which of the available devices. The execution time of the predicted placements is then used as the reward signal to optimize the parameters of the sequence-to-sequence model. Their method outperformed hand-crafted heuristics and traditional algorithmic methods.

Based on the intuition that, in HPC, executions at a smaller scale (in input problem size or node count) can be used to characterize executions at a larger scale, Marathe et al. [46] presented an effective method to identify high-performing application configurations when limited resources are available for collecting training performance data. More specifically, a deep learning technique augmented with domain transfer learning was used to capture the complex relationships between application-level and platform-level parameters and dependent metrics such as execution time. The model combined information from exhaustive observations collected at a smaller scale with limited observations collected at a larger target scale. The proposed approach can accurately predict performance in the regimes of interest to performance analysts while outperforming many traditional techniques.

Failure prediction. Today's large-scale supercomputers encounter failure on a daily basis. As they evolve to exascale systems, they are likely to experience even higher failure rates due to increased component count and operating frequency. The ability to accurately predict failure becomes crucial. Autonomous systems need to have a self-aware capability [42], so that they can take appropriate remedial action before failure.

A complement to the conventional checkpoint-restart strategy is failure avoidance, by which the occurrence of a fault is predicted and proactive measures are taken. Gainaru et al. [15] described the problems and limitations faced in developing an accurate failure predictor. The authors showed that a good way to achieve a viable solution for failure avoidance and overcome current limitations is to combine *signal analysis*, for shaping the normal behavior of each event type and of the whole system and characterizing the way faults affect them, and *data mining*, for analyzing the correlations between these behaviors.

Lan et al. [33] presented an automated mechanism for node-level anomaly identification in large-scale systems by using conventional statistical machine learning and data-mining techniques. Du et al. [14] proposed a deep neural network (DNN) model that uses the long short-term memory algorithm to model a system log as a natural language sequence. The model can automatically learn log patterns from normal execution and detect anomalies when log patterns deviate from the model trained from log data under normal execution.

Failure prediction and explanation for different types of computer systems, based on different kind of information, are essential steps toward an autonomous module that can proactively deal with failure in an autonomous way.

Power efficiency. As supercomputer systems get larger, power consumption become a necessary consideration [35] to ensure system stability, and a power-aware scheduler becomes crucial for controlling system power consumption while minimizing the impact on system utilization. Based on the key observation that HPC jobs have distinct power profiles, Wallace et al. [70] proposed a data-driven scheduling approach for controlling the power consumption of the entire system under any user-defined budget. The proposed approach actively observes, analyzes, and assesses power behaviors of the system and user jobs to guide scheduling decisions for power management. Since the dynamic learner is on-line and can react to changes in power profiles as jobs execute and can schedule subsequent jobs accordingly, it is generally applicable to other systems as long as the underlying platform has a power-monitoring facility for which data are available for specific portions of hardware.

The DeepMind team collaborated with the data center operations team at Google and used reinforcement learning to improve the air conditioning knobs [11]. The energy usage for cooling for the data center drops by about 30–35 percent after turning on the machine learning control system. This is another promising area where machine learning for computing systems can help.

2.3 Network

The research in autonomous aspects for networking broadly falls into three categories: routing, energy efficiency, and anomaly detection.

Routing. The research community has considered the application of artificial intelligence techniques to control and operate networks [48]. One notable example is the knowledge plane proposed by Clark et al. [9] in 2003. However, such techniques have not been extensively prototyped or deployed for practical use cases. Mestres et al. [48] explored the reasons for the lack of adoption and proposed that the rise of software-defined networking and network analytics will facilitate the adoption of artificial intelligence techniques in the context of network operation and control.

By taking advantage of recent breakthroughs in deep neural networks applied to reinforcement learning, Stampa et al. [65] designed and evaluated a deep reinforcement-learning-based agent that adapts automatically to current traffic conditions and proposes tailored configurations to optimize software-defined networking

routing (i.e., minimize the network delay). This work provides important operational advantages with respect to traditional optimization algorithms.

Energy. The bundle link is composed of several high-speed physical sublinks (SONET connections, Ethernet circuits, etc.) in order to make them work together as a virtual connection. This technique is widely used in current core networks to provide higher bandwidth and more resilience. Liu and Ramamurthy [40] proposed a dynamic, hybrid local heuristic threshold-based algorithm to achieve a tradeoff between energy savings and congestion risk. Specifically, the algorithm autonomously and dynamically shuts down and brings up sublinks and their attached ports according to the traffic demand or estimation, thereby greatly increasing the link utilization and saving a large amount of energy. Liu et al. [39] presented a multilayer energy-saving technique for optical core networks by powering off components in different layers of the network. Experiments on Internet2 show that an average of 88.66% of energy can be saved in an aggressive mode.

Anomaly detection. Nanda et al. [54] proposed using machine learning algorithms, trained on historical network attack data, to identify the potential malicious connections and potential attack destinations. Their experimental results showed that machine learning algorithms can help in defining security rules for software-defined networking controllers by accurately predicting the potential vulnerable host. They achieved an average prediction accuracy of 91.68% with a Bayesian network.

2.4 Storage

We organize the work on autonomous tuning of storage systems into two groups: performance and reliability.

Performance. Software-defined storage (SDS) is a new term for data storage software that provides policy-based provisioning and management of data storage independent of the underlying hardware. Tinedo et al. [19] presented the first SDS architecture whose core objective is to efficiently support multitenancy in object stores. The flexibility of developing control policy, such as by using the popular IFTTT (If-This-Then-That) service, provides an easier way to develop application-aware storage systems; automatic parameters tuning; and policy-based management, movement, and protection of systems and information.

In the big data era, the gap between storage performance and application I/O requirements is increasing [77]. I/O congestion caused by concurrent storage accesses from improperly scheduled applications can severely harm performance. Zhou et al. [77] presented an I/O-aware batch scheduling framework to alleviate the I/O congestion on petascale computing systems.

Paul et al. [58] proposed a data-driven load-balancing approach for the I/O servers of Lustre filesystems. The proposed global mapper runs on the metadata server, which gathers runtime statistics from key storage components on the I/O path via a publisher-subscriber model and applies Markov chain modeling and a minimum-cost maximum-flow algorithm to decide where data should be placed.

To find the optimal values of tunable parameters in Lustre-based storage systems, Li et al. [36] developed a modelless deep-reinforcement, learning-based, unsupervised parameter-tuning system driven by a deep neural network. Specifically, it takes periodic measurements of lustre filesystem's state and trains a DNN that uses Q-learning to suggest changes to the system's current parameter values. Their evaluation of a prototype on a Lustre filesystem demonstrates an increase in I/O throughput up to 45% at saturation point.

Reliability. One of the key requirements for storage systems is data reliability. Recent studies based on data from Facebook [49] and Google [64] report that 20–57% of solid-state drives experience at least one sector error. Sector errors are partial drive failures where individual sectors on a drive become unavailable. Sector errors occur at a high rate in both hard disk drives and solid state drives. Mahdisoltani et al. [45] explored a range of machine learning techniques and showed that sector errors can be predicted with high accuracy. Although the data in the affected sectors can be recovered through redundancy in the system (e.g. another drive in the same RAID), data are lost if the error is encountered during RAID reconstruction. An accurate prediction of sector error is valuable to avoid such data loss. Surprisingly, the sector error is predictable and the prediction is robust even when only little training data or only training data for a different drive model are available.

2.5 Data transfer node

Wide-area data transfer is central to distributed science [30]. Data transfer time directly influences workflow performance, and thus transfer throughput estimation is crucial for workflow scheduling and resource allocation [44]. Data transfer nodes (DTNs) [22] are compute systems dedicated to data transfers in science environments.

Wide area data transfers play an important role in many science applications, but rely on expensive infrastructure that often delivers disappointing performance in practice [41, 43] mostly due to the poor tuning of parameters. To this end, Liu et al. [42] proposed a smart data transfer node that uses deep reinforcement learning to maximize DTN aggregated throughput by dynamically tuning application-level parameters. They argued that such a system can identify transfer parameter values that achieve higher overall performance than simple heuristics can. Their results suggest that a knowledge engine that implements such methods can indeed guide a data transfer node to stable, sustained transfer performance.

3 OPEN ISSUES AND OPPORTUNITIES

The literature review in §2 suggests that a significant lag exists between the development of general autonomous computing concepts and the practical realization of even autonomous subsystems. Recent work suggests that machine learning methods have promise as a means of creating autonomous modules for computer systems. Expensive science instruments such as large telescopes and facilities such as colliders and synchrotrons also need autonomous capabilities. For example, at beamlines associated with light source facilities such as the Advanced Photon Source at Argonne National Laboratory, autonomous methods may be used to optimize usage

by increasing throughput and by proactively avoiding failures. Failure rates can depend on such factors as the data collection rate and the load placed on other supporting devices and on the software stack [18]. Scientific productivity can be improved by autonomously tuning experiment parameters (e.g., collection rate and s-ray dose) to maximize some combination of instrument availability and throughput.

Coexistence of several autonomous modules is required in order to handle multiple concerns, and it requires coordination mechanisms to avoid incoherent administration decisions, e.g., by using multi-agent game-based approach. Coordinating different autonomous modules to achieve higher-level goals is challenging and, to the best of our knowledge, has not been well investigated. Gueye et al. [20] investigated the use of control techniques in a coordination controller, in which they exercised synchronous programming that provides formal semantics and discrete controller synthesis to automate the construction of the controller.

Another issue is the limited availability (or lack) of annotated performance data for training machine learning models. Data about specific events, for example for training anomaly detection models, are also lacking. Collecting data is tedious and time consuming but essential in order to power machine learning models and foster autonomous science infrastructure research.

As systems scale toward exascale, many resources will become increasingly constrained [76]. While some resources have historically been allocated explicitly, others such as network bandwidth, I/O bandwidth, and power are not. As systems continue to evolve, we expect many such resources will need to be explicitly managed, making autonomous management of resources and coordinating autonomous resources to achieve higher-level goals even more important.

4 ARCHITECTURE PROPOSAL

We introduce architectural concepts that we believe will prove useful in future autonomous science infrastructures.

We note first that an autonomous science infrastructure will need to be designed that optimizes scientific impact. A difficulty here is that the nature of scientific impact and productivity may not always be well defined. Many authors have proposed and discussed methods for quantifying the impact of science [5, 6, 34, 59]. Each funding agency assesses impact in its own way [55, 61]. In the case of scientific facilities, funding agencies often set quantitative goals. For a genome facility, the goal might be "X" billion DNA bases sequenced in a year [26], while for a compute facility it might be "Y" billion core-hours delivered.

Operators of scientific facilities may then define subsidiary goals designed to increase their chances of meeting those high-level goals. For example, they might define a goal of "M"% availability for one system, "N"% utilization for another, and "O" operating hours for a third. Funding managers, facility operators, and system administrators currently use ad hoc methods and heuristics to set and achieve these goals.

Our proposed architecture is a small step toward making this process more autonomous than at present.

4.1 Science infrastructure

The science infrastructure (as depicted in Figure 2) is typically made up of scientific facilities funded by government agencies. government agencies fund these facilities, which in most cases are operated by other institutions. The supercomputers and other user facilities funded by the U.S. Department of Energy and National Science Foundation are examples of this model. The facilities in turn consist of a variety of resources such as high-performance computers, scientific instruments, storage systems, and computer networks.

4.2 Autonomous system

Figure 1 shows the architecture of an autonomous system. This architecture follows the architecture of an autonomous system proposed in a number of studies discussed in §2.1 (eg., MAPE-K loop [62]). The target system in Figure 1 can be a compute node, network, storage, instrument, or data transfer node dedicated for data movement over a wide-area network. The literature reviewed in §2.2–§2.5 focuses on adding different autonomous capabilities to these resources to form an autonomous science infrastructure.

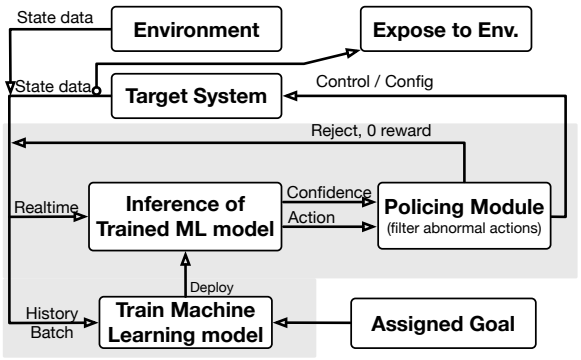


Figure 1: Architecture of an autonomous system. The shaded area is the "Knowledge Engine (KE)" that provides autonomous capabilities to the "Target System."

4.3 Knowledge engine

The knowledge engine enables a system to achieve a specified goal, for example, maximize performance or minimize power consumption, in an autonomous fashion. The state/performance data that represent the state of the target system are collected by monitoring modules. In real time, these data are the input of the knowledge engine, and the output is the control command (decision). The pair of control command and the corresponding change of system state can reflect the relationship between control command and system state transition. Thus, the state-command-state data can be used to train a machine learning model (e.g., a reinforcement machine learning model such as Q-learning [71] or policy gradient [37]) so that the machine learning model captures the relationship and makes better decisions in the next runs. Specifically for reinforcement learning that learns what to do—how to map situations to actions—so as to maximize a numerical reward [66], there is a reward function,

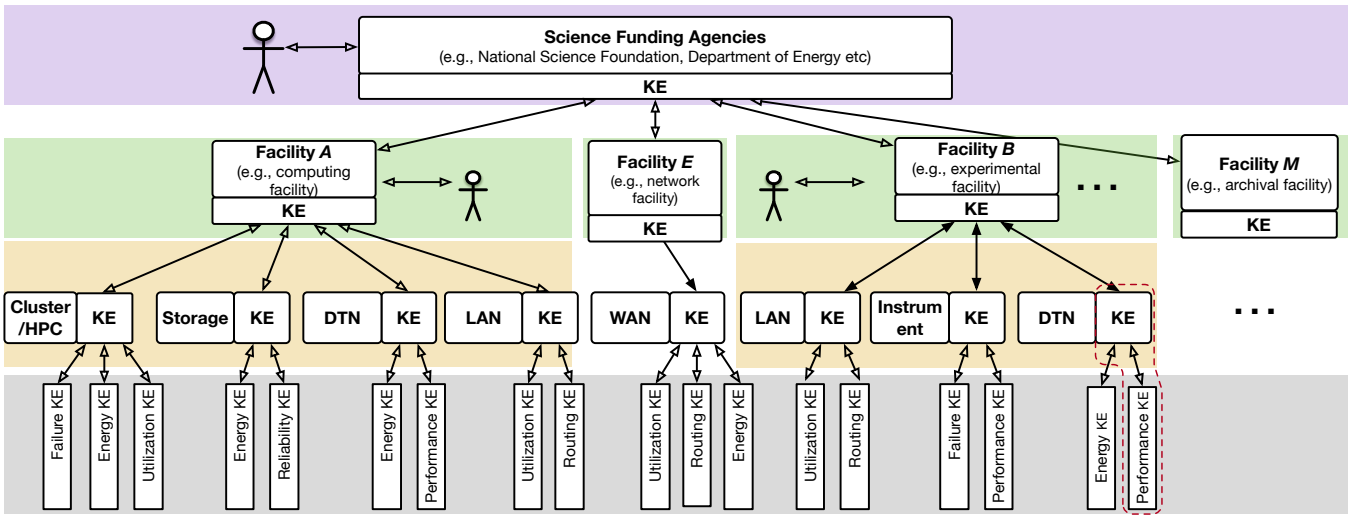


Figure 2: Architecture of autonomous science infrastructure. Each node is equipped with a knowledge engine to make optimal decisions. The architecture of each node is shown in Figure 1. For example, the red dotted box is the autonomous module that autotunes data transfer parameters to maximize the aggregated throughput of DTN, as presented in [42].

to maximize, that quantify rewards the system gets from the state transition.

The knowledge engine can run on the target system itself (if it has sufficient computing resources) or on an edge computer that is attached to the target system. The training and inferencing modules of KE can be separated, and the training (which is typically more compute intensive than inferencing) can be done on a remote computer; inferencing can be done on the target system or on a lightweight special-purpose machine learning accelerator (e.g., the recently announced Intel Movidius neural computing stick [51]) attached to the target system.

Policing modules with static rules (e.g., reference interval) will verify decisions before applying to the actual system and will reject abnormal decisions. In order to let the knowledge engine know that the decision is abnormal, so as to discourage such decisions, the policing module will feed back a zero reward for the decision to the training module.

We note that Figure 1 shows only a high-level architecture. Although the architecture is generic for different systems, the KE model will be different for different systems. Even the same type of system (e.g., storage) may have a different KE for different products (e.g., GPFS, Lustre). In fact, even for the same product, we may need different KEs for different goals.

4.4 Autonomous science infrastructure

As discussed in §4.2, Figure 2 shows the hierarchical structure of the science infrastructure. Government agencies create scientific facilities with an overarching goal to advance science. These agencies set some higher-level goals and guidelines for each facility, but the facilities are typically operated by other institutions such as a university, supercomputer center, or a national laboratory. Within the guidelines of the funding agencies, these institutions operate the scientific facilities to achieve the higher-level goals set by the funding agencies and other goals of national interest deemed important by them.

Our vision for autonomous science infrastructure is one in which there is a hierarchy of autonomous elements that takes goals from their parent and autonomously tune either the goals of their children (applies to nonleaf nodes) or their own tunable parameters (applies to leaf nodes) to achieve their goals. Administrators can set goals as well, and these will take higher priority than the ones set by the parent. The topmost autonomous element (one without a parent) takes goals only from its administrator. We expect that administrators will set goals for the autonomous elements at the root of each administrative domain.

Our vision of the architecture for an autonomous science infrastructure is presented in Figure 2. We represent the infrastructure as a tree, with the funding agency as an autonomous element at the root; facilities as autonomous elements at the next level below; systems such as high-performance computers, storage systems, networks, and instruments as autonomous elements under facilities (note that facilities are heterogeneous); and the modules that achieve a low-level goal, such as "99.9% uptime for a storage system," as autonomous elements at the leaf level. The autonomous elements are architected as shown in Figure 1. The 'Target System" for the KE of the autonomous elements at leaf nodes is the physical system (such as a high-performance computer, storage system, or network). Thus, the "Action" output of the KE on these nodes is a new set of values for the configuration parameters of the system. The "Target System" for the KE of the autonomous elements at nonleaf nodes is one or more of their children's KE. Thus, the "Action" output of the KE on these nodes is a new set of goals for KE(s) on their child(ren). Each autonomous element operates autonomously to achieve the goal (set by a parent element or a human administrator). We note that the autonomous elements can be designed to support multiple tiers of goals (as proposed in [13]) so that they can satisfy a

weaker goal if it is not possible to saitsfy the desired goal in certain circumstances (e.g., due to failure of components).

We use the example of an autonomous data transfer node to illustrate how the proposed hierarchical autonomous system would work in practice. Figure 3 shows an autonomous DTN with two autonomous modules—one that autonomously achieves a performance goal (Figure 4a) and one that autonomously achieves a power consumption goal (Figure 4b). Let's say that the goal of DTN is to minimize the price-performance ratio. To achieve such a goal, it assigns a power consumption goal to the KE shown in Figure 4b and data transfer performance goal to KE shown in Figure 4a. The KE of the power module strives to achieve its goal autonomously by learning the relationship between the system parameters that affect the power consumption, environmental conditions, and the amount of power consumed. The KE of the performance module does a similar job to achieve its goal. We note that minimizing power consumption and maximizing performance are two conflicting goals. The KE of the DTN needs to learn the tradeoff between them for varying environmental conditions in order to achieve its goal.

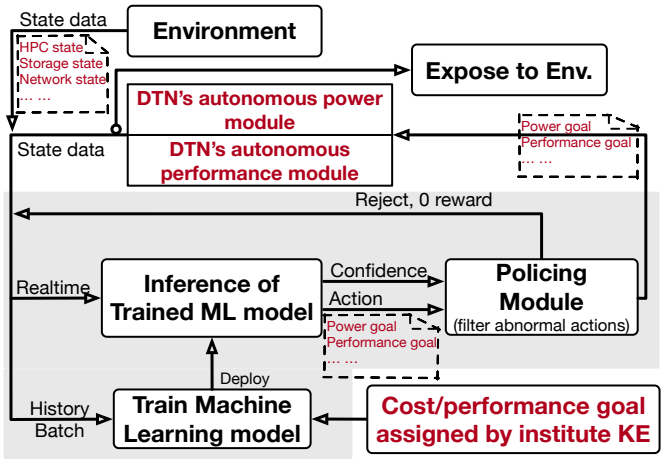
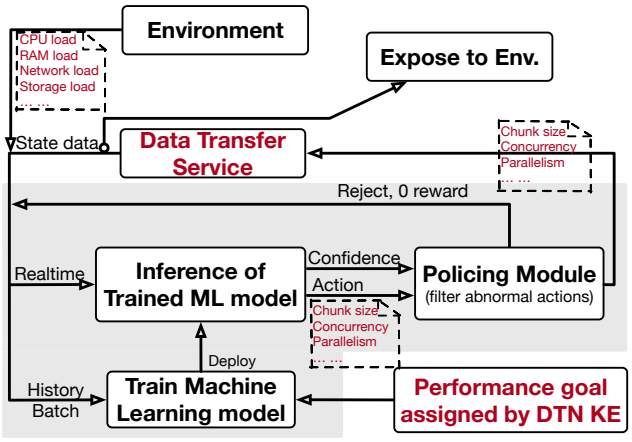
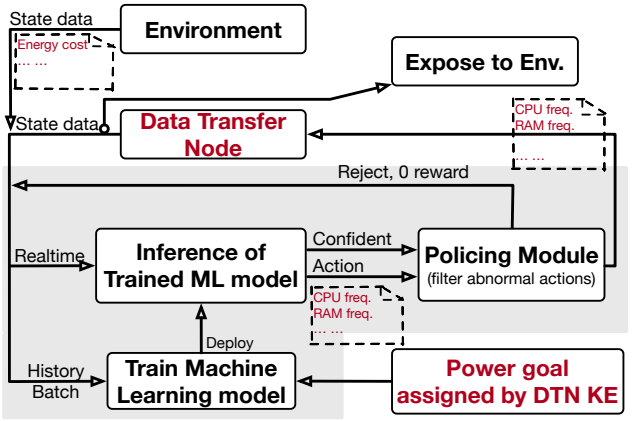


Figure 3: Illustration of a hierarchical autonomous system showing how an autonomous data transfer node achieves its goal by tuning the goals of the underlying autonomous modules (shown in Figure 4a and Figure 4b).



(a) Autonomous performance module for data transfer nodes.



(b) Autonomous power module for data transfer nodes.

Figure 4: The underlying autonomous modules of an autonomous DTN as shown in Figure 3. These modules achieve their goal by tuning the parameters of hardware/software-tools on the data transfer node.

5 CONCLUSION

Autonomous systems have been talked about for a long time, and significant progress has occurred in areas such as self-driving vehicles. Practical realization of autonomous systems in scientific computing environments has been limited, however. Nevertheless, with the latest advancements in machine learning, researchers have expressed considerable interest in applying machine learning to autonomously optimize individual aspects of a system. We reviewed recent work in this space for systems that form a key part of scientific computing infrastructure. Challenges remain in coordinating individual autonomous modules to achieve higher-level goals. Building on previous work on autonomous computing systems, we proposed a hierarchical architecture for an autonomous science infrastructure that spans administrative boundaries. We view our work as a step toward making autonomous systems practical.

ACKNOWLEDGMENT

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357 and the DOE AMASE project fund by SmartHPCC program managed by Thomas Ndousse-Fetter. We also would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed

Systems. *CoRR* abs/1603.04467 (2016). arXiv:1603.04467 http://arxiv.org/abs/1603.04467

[2] Nazim Agoulmine, Sasitharan Balasubramaniam, Dmitri Botvich, John Strassner, Elyes Lehtihet, and William Donnelly. 2006. Challenges for autonomic network management. In *1st IEEE International Workshop on Modelling Autonomic Communications Environments*.

[3] Mark Allman, Vern Paxson, and Ethan Blanton. 2009. *TCP congestion control*. Technical Report. https://doi.org/10.17487/RFC5681

[4] Peter Bodík, Rean Griffith, Charles Sutton, Armando Fox, Michael Jordan, and David Patterson. 2009. Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing (HotCloud'09)*. USENIX Association, Berkeley, CA, USA, Article 12. http://dl.acm.org/citation.cfm?id=1855533.1855545

[5] Lutz Bornmann. 2012. Measuring the societal impact of research: research is less and less assessed on scientific impact alone – we should aim to quantify the increasingly important contributions of science to society. *EMBO reports* 13, 8 (2012), 673–676. https://doi.org/10.1038/embor.2012.99

[6] Philip Campbell and Michelle Grayson. 2014. Assessing science. *Nature* 511, S49 (2014). https://doi.org/10.1038/511S49a

[7] Danilo Carastan-Santos and Raphael Y. de Camargo. 2017. Obtaining Dynamic Scheduling Policies with Simulation and Machine Learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 32, 13 pages. https://doi.org/10.1145/3126908.3126955

[8] Giuliano Casale. 2017. Accelerating Performance Inference over Closed Systems by Asymptotic Methods. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 1, Article 17 (2017), 36 pages. https://doi.org/10.1145/3084445

[9] David D. Clark, Craig Partridge, J. Christopher Ramming, and John T. Wroclawski. 2003. A Knowledge Plane for the Internet. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*. ACM, New York, NY, USA, 3–10. https://doi.org/10.1145/863955.863957

[10] Jeff Dean. 2017. *Machine Learning for Systems and Systems for Machine Learning*. http://learningsys.org/nips17/assets/slides/dean-nips17.pdf.

[11] Deepmind. 2018 (accessed March 3, 2018). *DeepMind AI Reduces Google Data Centre Cooling Bill by 40%*. https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40.

[12] Peter A. Dinda and David R. O'Hallaron. 2000. Host Load Prediction Using Linear Models. *Cluster Computing* 3, 4 (Oct. 2000), 265–280. https://doi.org/10.1023/A:1019048724544

[13] Nicolas D'Ippolito, Víctor Braberman, Jeff Kramer, Jeff Magee, Daniel Sykes, and Sebastian Uchitel. 2014. Hope for the Best, Prepare for the Worst: Multi-tier Control for Adaptive Systems. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 688–699. https://doi.org/10.1145/2568225.2568264

[14] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 1285–1298. https://doi.org/10.1145/3133956.3134015

[15] Ana Gainaru, Franck Cappello, Marc Snir, and William Kramer. 2013. Failure prediction for HPC systems and applications: Current situation and open issues. *The International Journal of High Performance Computing Applications* 27, 3 (2013), 273–282. https://doi.org/10.1177/1094342013488258 arXiv:https://doi.org/10.1177/1094342013488258

[16] A. G. Ganek and T. A. Corbi. 2003. The dawning of the autonomic computing era. *IBM Systems Journal* 42, 1 (2003), 5–18. https://doi.org/10.1147/sj.421.0005

[17] E. Gaussier, D. Glesser, V. Reis, and D. Trystram. 2015. Improving backfilling by using machine learning to predict running times. In *SC15: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–10. https://doi.org/10.1145/2807591.2807646

[18] Daniel Gewirth. [n. d.]. The HKL manual. ([n. d.]). https://dasher.wustl.edu/bio5325/reading/hkl-manual.pdf

[19] Raúl Gracia-Tinedo, Josep Sampé, Edgar Zamora, Marc Sánchez-Artigas, Pedro García-López, Yosef Moatti, and Eran Rom. 2017. Crystal: Software-Defined Storage for Multi-Tenant Object Stores. In *15th USENIX Conference on File and Storage Technologies*. USENIX Association, Santa Clara, CA, 243–256. https://www.usenix.org/conference/fast17/technical-sessions/presentation/gracia-tinedo

[20] Soguy Mak-KarÃľ Gueye, NoÃńl De Palma, Ã Ľric Rutten, Alain Tchana, and Nicolas Berthier. 2014. Coordinating self-sizing and self-repair managers for multi-tier systems. *Future Generation Computer Systems* 35 (2014), 14–26. https://doi.org/10.1016/j.future.2013.12.037 Special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan.

[21] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. 2013. Self-adaptive Workload Classification and Forecasting for Proactive Resource Provisioning. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE '13)*. ACM, New York, NY, USA, 187–198. https://doi.org/10.1145/2479871.2479899

[22] https://www.es.net. [n. d.]. *Science DMZ: Data Transfer Nodes*. https://fasterdata.es.net/science-dmz/DTN/.

[23] Markus C Huebscher and Julie A McCann. 2008. A survey of autonomic computing–degrees, models, and applications. *Comput. Surveys* 40, 3 (2008), 7.

[24] Hameed Hussain, Saif Ur Rehman Malik, Abdul Hameed, Samee Ullah Khan, Gage Bickler, Nasro Min-Allah, Muhammad Bilal Qureshi, Limin Zhang, Wang Yongji, Nasir Ghani, Joanna Kolodziej, Albert Y. Zomaya, Cheng-Zhong Xu, Pavan Balaji, Abhinav Vishnu, Fredric Pinel, Johnatan E. Pecero, Dzmitry Kliazovich, Pascal Bouvry, Hongxiang Li, Lizhe Wang, Dan Chen, and Ammar Rayes. 2013. A survey on resource allocation in high performance distributed computing systems. *Parallel Comput.* 39, 11 (2013), 709–736. https://doi.org/10.1016/j.parco.2013.09.009

[25] IBM. 2018 (accessed April 3, 2018). An architectural blueprint for autonomic computing. (2018 (accessed April 3, 2018)). http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf.

[26] JGI: Joint Genome Institute. [n. d.]. *DOE Metrics/Statistics*. https://jgi.doe.gov/our-projects/statistics/.

[27] J. O. Kephart. 2005. Research challenges of autonomic computing. In *27th International Conference on Software Engineering*. 15–22. https://doi.org/10.1109/ICSE.2005.1553533

[28] J. O. Kephart and D. M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (Jan 2003), 41–50. https://doi.org/10.1109/MC.2003.1160055

[29] Jeffrey O Kephart and David M Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (2003), 41–50.

[30] Rajkumar Kettimuthu, Zhengchun Liu, David Wheelerd, Ian Foster, Katrin Heitmann, and Franck Cappello. 2017. Transferring a Petabyte in a Day. In *4th International Workshop on Innovating the Network for Data Intensive Science*. 10.

[31] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey. 2016. Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. 1–10. https://doi.org/10.1109/CLOUD.2016.0011

[32] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2017. The Case for Learned Index Structures. *arXiv preprint arXiv:1712.01208* (2017).

[33] Zhiling Lan, Ziming Zheng, and Yawei Li. 2010. Toward Automated Anomaly Identification in Large-Scale Systems. *IEEE Trans. Parallel Distrib. Syst.* 21, 2 (Feb. 2010), 174–187. https://doi.org/10.1109/TPDS.2009.52

[34] Julia Lane. 2009. Assessing the Impact of Science Funding. *Science* 324, 5932 (2009), 1273–1275. https://doi.org/10.1126/science.1175335

[35] Bo Li, Edgar A. León, and Kirk W. Cameron. 2017. COS: A Parallel Performance Model for Dynamic Variations in Processor Speed, Memory Speed, and Thread Concurrency. In *26th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '17)*. ACM, New York, NY, USA, 155–166. https://doi.org/10.1145/3078597.3078601

[36] Yan Li, Kenneth Chang, Oceane Bel, Ethan L. Miller, and Darrell D. E. Long. 2017. CAPES: Unsupervised Storage Performance Tuning Using Neural Network-based Deep Reinforcement Learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 42, 14 pages. https://doi.org/10.1145/3126908.3126951

[37] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015). http://arxiv.org/abs/1509.02971

[38] Marin Litoiu, Mary Shaw, Gabriel Tamura, Norha M. Villegas, Hausi A. Müller, Holger Giese, Romain Rouvoy, and Eric Rutten. 2017. What Can Control Theory Teach Us About Assurances in Self-Adaptive Software Systems?. In *Software Engineering for Self-Adaptive Systems III. Assurances*, Rogério de Lemos, David Garlan, Carlo Ghezzi, and Holger Giese (Eds.). Springer International Publishing, Cham, 90–134.

[39] L. Liu, S. E. Alaoui, and B. Ramamurthy. 2014. Multi-layer energy savings in optical core networks. In *IEEE International Conference on Advanced Networks and Telecommuncations Systems*. 1–3. https://doi.org/10.1109/ANTS.2014.7057279

[40] L. Liu and B. Ramamurthy. 2011. A dynamic local method for bandwidth adaptation in bundle links to conserve energy in core networks. In *5th IEEE International Conference on Advanced Telecommunication Systems and Networks*. 1–6. https://doi.org/10.1109/ANTS.2011.6163644

[41] Zhengchun Liu, Prasanna Balaprakash, Rajkumar Kettimuthu, and Ian Foster. 2017. Explaining Wide Area Data Transfer Performance. In *26th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '17)*. ACM, New York, NY, USA, 167–178. https://doi.org/10.1145/3078597.3078605

[42] Zhengchun Liu, Rajkumar Kettimuthu, Ian Foster, and Peter H. Beckman. 2017. Towards a Smart Data Transfer Node. In *4th International Workshop on Innovating the Network for Data Intensive Science*. 10.

[43] Zhengchun Liu, Rajkumar Kettimuthu, Ian Foster, and Nageswara S.V. Rao. 2018. Cross-geography Scientific Data Transfer Trends and User Behavior Patterns. In *27th ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC '18)*. ACM, New York, NY, USA, 12.

[44] Zhengchun Liu, Rajkumar Kettimuthu, Sven Leyffer, Prashant Palkar, and Ian Foster. 2017. A Mathematical Programming- and Simulation-Based Framework to Evaluate Cyberinfrastructure Design Choices. In *IEEE 13th International Conference on e-Science*. 148–157. https://doi.org/10.1109/eScience.2017.27

[45] Farzaneh Mahdisoltani, Ioan Stefanovici, and Bianca Schroeder. 2017. Proactive error prediction to improve storage system reliability. In *USENIX Annual Technical Conference*. USENIX Association, Santa Clara, CA, 391–402. https://www.usenix.org/conference/atc17/technical-sessions/presentation/mahdisoltani

[46] Aniruddha Marathe, Rushil Anirudh, Nikhil Jain, Abhinav Bhatele, Jayaraman Thiagarajan, Bhavya Kailkhura, Jae-Seung Yeom, Barry Rountree, and Todd Gamblin. 2017. Performance Modeling Under Resource Constraints Using Deep Transfer Learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 31, 12 pages. https://doi.org/10.1145/3126908.3126969

[47] A. Matsunaga and J. A. B. Fortes. 2010. On the Use of Machine Learning to Predict the Time and Resources Consumed by Applications. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. 495–504. https://doi.org/10.1109/CCGRID.2010.98

[48] Albert Mestres, Alberto Rodríguez-Natal, Josep Carner, Pere Barlet-Ros, Eduard Alarcón, Marc Solé, Victor Muntés, David Meyer, Sharon Barkai, Mike J. Hibbett, Giovani Estrada, Khaldun Maruf, Florin Coras, Vina Ermagan, Hugo Latapie, Chris Cassar, John Evans, Fabio Maino, Jean C. Walrand, and Albert Cabellos. 2016. Knowledge-Defined Networking. *CoRR* abs/1606.06222 (2016). arXiv:1606.06222 http://arxiv.org/abs/1606.06222

[49] Justin Meza, Qiang Wu, Sanjev Kumar, and Onur Mutlu. 2015. A Large-Scale Study of Flash Memory Failures in the Field. In *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '15)*. ACM, New York, NY, USA, 177–190. https://doi.org/10.1145/2745844.2745848

[50] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. 2017. Device Placement Optimization with Reinforcement Learning. *CoRR* abs/1706.04972 (2017). arXiv:1706.04972 http://arxiv.org/abs/1706.04972

[51] Movidius. 2018 (accessed April 3, 2018). Intel Movidius Neural Compute Stick. (2018 (accessed April 3, 2018)). https://developer.movidius.com/.

[52] Steven S Muchnick. 1997. *Advanced compiler design implementation*. Morgan Kaufmann.

[53] M. R. Nami and K. Bertels. 2007. A Survey of Autonomic Computing Systems. In *3rd International Conference on Autonomic and Autonomous Systems*. 26–26. https://doi.org/10.1109/CONIELECOMP.2007.48

[54] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang. 2016. Predicting network attack patterns in SDN using machine learning approach. In *IEEE Conference on Network Function Virtualization and Software Defined Networks*. 167–172. https://doi.org/10.1109/NFV-SDN.2016.7919493

[55] National Research Council. 1998. *Assessing the Value of Research in the Chemical Sciences*. National Academies Press. https://books.google.com/books?id=F0-2Nn3llYQC

[56] Shuangcheng Niu, Jidong Zhai, Xiaosong Ma, Xiongchao Tang, and Wenguang Chen. 2013. Cost-effective Cloud HPC Resource Provisioning by Building Semi-elastic Virtual Clusters. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13)*. ACM, New York, NY, USA, Article 56, 12 pages. https://doi.org/10.1145/2503210.2503236

[57] Manish Parashar and Salim Hariri. 2005. Autonomic Computing: An Overview. In *Unconventional Programming Paradigms*, Jean-Pierre Banâtre, Pascal Fradet, Jean-Louis Giavitto, and Olivier Michel (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 257–269.

[58] A. K. Paul, A. Goyal, F. Wang, S. Oral, A. R. Butt, M. J. Brim, and S. B. Srinivasa. 2017. I/O load balancing for big data HPC applications. In *2017 IEEE International Conference on Big Data (Big Data)*. 233–242. https://doi.org/10.1109/BigData.2017.8257931

[59] Teresa Penfield, Matthew J. Baker, Rosa Scoble, and Michael C. Wykes. 2014. Assessment, evaluations, and definitions of research impact: A review. *Research Evaluation* 23, 1 (2014), 21–32. https://doi.org/10.1093/reseval/rvt021

[60] Xiaoqi Ren, Ganesh Ananthanarayanan, Adam Wierman, and Minlan Yu. 2015. Hopper: Decentralized Speculation-aware Cluster Scheduling at Scale. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 379–392. https://doi.org/10.1145/2829988.2787481

[61] Rosalie Ruegg and Gretchen Jordan. 2007. *Overview of evaluation methods for R&D programs*. Technical Report. U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy.

[62] Eric Rutten, Nicolas Marchand, and Daniel Simon. 2015. *Feedback Control as MAPE-K loop in Autonomic Computing*. Research Report RR-8827. INRIA Sophia Antipolis - Méditerranée ; INRIA Grenoble - Rhône-Alpes. https://hal-lirmm.ccsd.cnrs.fr/lirmm-01241594 draft soumis à LNCS.

[63] Mazeiar Salehie and Ladan Tahvildari. 2005. Autonomic Computing: Emerging Trends and Open Problems. *SIGSOFT Softw. Eng. Notes* 30, 4 (May 2005), 1–7. https://doi.org/10.1145/1082983.1083082

[64] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. 2016. Flash Reliability in Production: The Expected and the Unexpected. In *14th USENIX Conference on File and Storage Technologies*. USENIX Association, Santa Clara, CA, 67–80. https://www.usenix.org/conference/fast16/technical-sessions/presentation/schroeder

[65] Giorgio Stampa, Marta Arias, David Sanchez-Charles, Victor Muntés-Mulero, and Albert Cabellos. 2017. A Deep-Reinforcement Learning Approach for Software-Defined Networking Routing Optimization. *CoRR* abs/1709.07080 (2017). arXiv:1709.07080 http://arxiv.org/abs/1709.07080

[66] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT Press Cambridge.

[67] Andrew S Tanenbaum. 2009. *Modern operating system*. Pearson Education, Inc.

[68] G. Tesauro. 2007. Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies. *IEEE Internet Computing* 11, 1 (Jan 2007), 22–30. https://doi.org/10.1109/MIC.2007.21

[69] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, and Steve R. White. 2004. A Multi-Agent Systems Approach to Autonomic Computing. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '04)*. IEEE Computer Society, Washington, DC, USA, 464–471. https://doi.org/10.1109/AAMAS.2004.23

[70] Sean Wallace, Xu Yang, Venkatram Vishwanath, William E. Allcock, Susan Coghlan, Michael E. Papka, and Zhiling Lan. 2016. A Data Driven Scheduling Approach for Power Management on HPC Systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*. IEEE Press, Piscataway, NJ, USA, Article 56, 11 pages. http://dl.acm.org/citation.cfm?id=3014904.3014979

[71] Christopher Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3-4 (1992), 279–292.

[72] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess, and J. O. Kephart. 2004. An architectural approach to autonomic computing. In *International Conference on Autonomic Computing*. 2–9. https://doi.org/10.1109/ICAC.2004.1301340

[73] Steve R White, James E Hanson, Ian Whalley, David M Chess, Alla Segal, and Jeffrey O Kephart. 2006. Autonomic computing: Architectural approach and prototype. *Integrated Computer-Aided Engineering* 13, 2 (2006), 173–188.

[74] Neeraja J. Yadwadkar, Ganesh Ananthanarayanan, and Randy Katz. 2014. Wrangler: Predictable and Faster Jobs Using Fewer Resources. In *Proceedings of the ACM Symposium on Cloud Computing (SOCC '14)*. ACM, New York, NY, USA, Article 26, 14 pages. https://doi.org/10.1145/2670979.2671005

[75] Pengfei Zheng and Benjamin C. Lee. 2018. Hound: Causal Learning for Datacenter-scale Straggler Diagnosis. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, Vol. 2. 1–36. https://doi.org/10.1145/3179420

[76] Zhou, Xu Yang, Zhiling Lan, Paul Rich, Wei Tang, Vitali Morozov, and Narayan Desai. 2016. Improving Batch Scheduling on Blue Gene/Q by Relaxing Network Allocation Constraints. *IEEE Trans. Parallel Distrib. Syst.* 27, 11 (Nov. 2016), 3269–3282. https://doi.org/10.1109/TPDS.2016.2528247

[77] Z. Zhou, X. Yang, D. Zhao, P. Rich, W. Tang, J. Wang, and Z. Lan. 2015. I/O-Aware Batch Scheduling for Petascale Computing Systems. In *IEEE International Conference on Cluster Computing*. 254–263. https://doi.org/10.1109/CLUSTER.2015.45